In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [4]:

```
Data=pd.read_excel('Housing.xlsx')
```

In [5]:

```
Data.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
0	-122.23	37.88	41	880	129.0	322	12
1	-122.22	37.86	21	7099	1106.0	2401	113
2	-122.24	37.85	52	1467	190.0	496	17
3	-122.25	37.85	52	1274	235.0	558	21
4	-122.25	37.85	52	1627	280.0	565	25

In [6]:

```
Label=Data.iloc[:,[-1]]
Feature=Data.iloc[:,[0,1,2,3,4,5,6,7,8]]
```

In [7]:

```
Feature['total_bedrooms'].fillna(Feature.total_bedrooms.mean(),inplace=True)
Feature.total_bedrooms=Feature.total_bedrooms.astype(int)
```

```
C:\Users\Jyothish\Anaconda3\lib\site-packages\pandas\core\generic.py:613
0: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-do
cs/stable/indexing.html#indexing-view-versus-copy
  self._update_inplace(new_data)
C:\Users\Jyothish\Anaconda3\lib\site-packages\pandas\core\generic.py:509
6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-do
cs/stable/indexing.html#indexing-view-versus-copy
  self[name] = value
```

In [8]:

```
Feature=Feature.values
Label=Data.iloc[:,[-1]]. values
```

In [9]:

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
OceanEncoder=LabelEncoder()
Feature[:,8]=OceanEncoder.fit_transform(Feature[:,8])
```

In [10]:

```
OceanOHE=OneHotEncoder(categorical_features=[8])
Feature=OceanOHE.fit_transform(Feature).toarray()
```

```
C:\Users\Jyothish\Anaconda3\lib\site-packages\sklearn\preprocessing\_enc
oders.py:371: FutureWarning: The handling of integer data will change in
version 0.22. Currently, the categories are determined based on the rang
e [0, max(values)], while in the future they will be determined based on
the unique values.
If you want the future behaviour and silence this warning, you can speci
fy "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the
categories to integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
C:\Users\Jyothish\Anaconda3\lib\site-packages\sklearn\preprocessing\_enc
oders.py:392: DeprecationWarning: The 'categorical_features' keyword is
deprecated in version 0.20 and will be removed in 0.22. You can use the
ColumnTransformer instead.
  "use the ColumnTransformer instead.", DeprecationWarning)
```

In [11]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(Feature,Label,test_size=0.2,random_st
ate=40)
```

In [12]:

```
from sklearn.preprocessing import StandardScaler
SC_Scale=StandardScaler()
Feature=SC_Scale.fit_transform(Feature)
Feature
```

Out[12]:

```
array([[-0.89115574, -0.68188905, -0.01556621, ..., -0.9744286 ,
        -0.97703285,  2.34476576],
       [-0.89115574, -0.68188905, -0.01556621, ...,  0.86143887,
         1.66996103,  2.33223796],
       [-0.89115574, -0.68188905, -0.01556621, ..., -0.82077735,
        -0.84363692,  1.7826994 ],
       ...,
       [-0.89115574,  1.46651424, -0.01556621, ..., -0.3695372 ,
        -0.17404163, -1.14259331],
       [-0.89115574,  1.46651424, -0.01556621, ..., -0.60442933,
        -0.39375258, -1.05458292],
       [-0.89115574,  1.46651424, -0.01556621, ..., -0.03397701,
         0.07967221, -0.78012947]])
```

In [13]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(Feature,Label,test_size=0.2,random_st
ate=928)
```

In [14]:

```
from sklearn.linear_model import LinearRegression
Model = LinearRegression()
Model.fit(X_train,Y_train)
```

Out[14]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [15]:

```
print("Training Score: ",Model.score(X_train,Y_train))
print("Testing Score: ",Model.score(X_test,Y_test))
```

```
Training Score:  0.634816811624079
Testing Score:  0.6872134425749434
```

In [16]:

```
Y_pre = Model.predict(Feature)
```

In [17]:

```
from sklearn.metrics import mean_squared_error
Mean_value=mean_squared_error(Label,Y_pre)
Mean_value
```

Out[17]:

```
4721770533.973778
```

In [18]:

```
from sklearn.tree import DecisionTreeRegressor
Model1=DecisionTreeRegressor(max_depth=3)
Model1.fit(X_train,Y_train)
```

Out[18]:

```
DecisionTreeRegressor(criterion='mse', max_depth=3, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [19]:

```
Model1.score(X_train,Y_train)
```

Out[19]:

```
0.562049235762236
```

In [20]:

```
Model1.score(X_test,Y_test)
```

Out[20]:

```
0.6012921337587399
```

In [21]:

```
from sklearn.ensemble import RandomForestRegressor
Model2=RandomForestRegressor(n_estimators=61,max_depth=6)
Model2.fit(X_train,Y_train)
```

```
C:\Users\Jyothish\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: D
ataConversionWarning: A column-vector y was passed when a 1d array was e
xpected. Please change the shape of y to (n_samples,), for example using
ravel().
  This is separate from the ipykernel package so we can avoid doing impo
rts until
```

Out[21]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=6,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=61, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=Fal
se)
```

In [22]:

```
Model2.score(X_train,Y_train)
```

Out[22]:

```
0.7056343503777647
```

In [23]:

```
Model2.score(X_test,Y_test)
```

Out[23]:

```
0.7214472486584113
```

In [24]:

```
Feature=Data[['median_income']].values
Label=Data[['median_house_value']].values
```

In [25]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(Feature,Label,test_size=0.2,random_st
ate=928)
```

In [26]:

```
from sklearn.linear_model import LinearRegression
Model = LinearRegression()
Model.fit(X_train,Y_train)
```

Out[26]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [27]:

```
print("Training Score: ",Model.score(X_train,Y_train))
print("Testing Score: ",Model.score(X_test,Y_test))
```

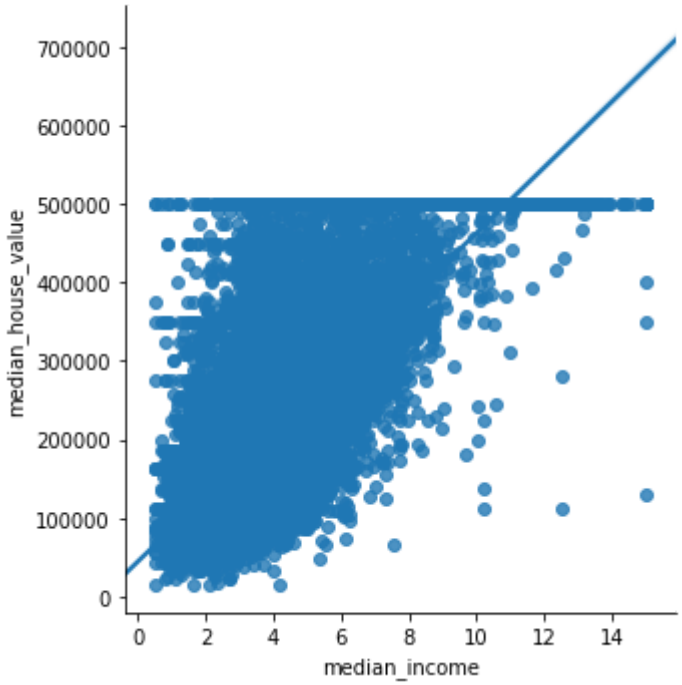
```
Training Score:  0.46374666206693066
Testing Score:  0.5116182192037121
```

In [28]:

```
sns.lmplot(x='median_income',y='median_house_value',data=Data)
```

Out[28]:

```
<seaborn.axisgrid.FacetGrid at 0x2179eb1c780>
```



In [ ]: